PROJECT REPORT
ON
TRANSFORMATION TOOL FOR TRANSFORMING PLATFORM INDEPENDENT MODEL INTO PLATFORM SPECIFIC MODEL

SUBMITTED BY

Nakul Tandon **(061260)**
Abhishek Mittal **(061202)**

UNDER GUIDANCE OF
Mr. Yashwant Singh (Lect. C.S.E Deptt.)

DECEMBER-2009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

TO WHOM IT MAY CONCERN

This is to certify that project entitled "TRANSFORMATION TOOL FOR TRANSFORMING PLATFORM INDEPENDENT MODEL INTO PLATFORM SPECIFIC MODEL" is work done by Nakul Tandon and Abhishek Mittal in partial fulfillment of requirement for the award of Bachelor of Engineering in Computer Science and Engineering. The project has been carried out under my direct supervision and guidance. This report or similar report on the topic has not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature of the Guide
Mr. Yashwant Singh
(Lecturer)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

AKNOWLEDGEMENT

I acknowledge a great sense of gratitude to Mr Yashwant Singh (Lect. Computer Science & Engineering) being my project guide had steadying influence on the project. He always managed to pick the broken links which helped me a lot in my project. He played a mojar role in the layout of this project.

There are no words through which I can thank Neha Arora(Management Engg. , Infosys Chandigarh) for her constant support and encouragement that I received constantly throughout the project. It is a mtter of fact that her support made us achieve the greatest height in developing this project which was not possible otherwise.

Lastly a special thanks to my friend Raghav Sadhir for his support through yet another long and contention consuming project.

Nakul Tandon (061260)
Abhishek Mittal (061202)

ABSTRACT

In today's world the basic need of every software developer is to provide the customer with software which can suit his needs to the best. Also, the time has come to get rid of age old software development models where heavy documentations were needed.
Our project is designed in such a way that the developer will entirely shift his focus on customer's needs thus resulting in a much improved and dynamic software development procedure.
Our project will not only benefit the user but will also be of great help for the developer who will have to tackle less issues related to the platform specifications and can thus easily shift his focus on the model or the design of the required software.

TABLE OF CONTENTS

PROBLEM STATEMENT

Developing a transformation tool for transforming Platform Independent Model (PIM) into Platform Specific Model (PSM).

OBJECTIVE AND SCOPE OF THE PROJECT

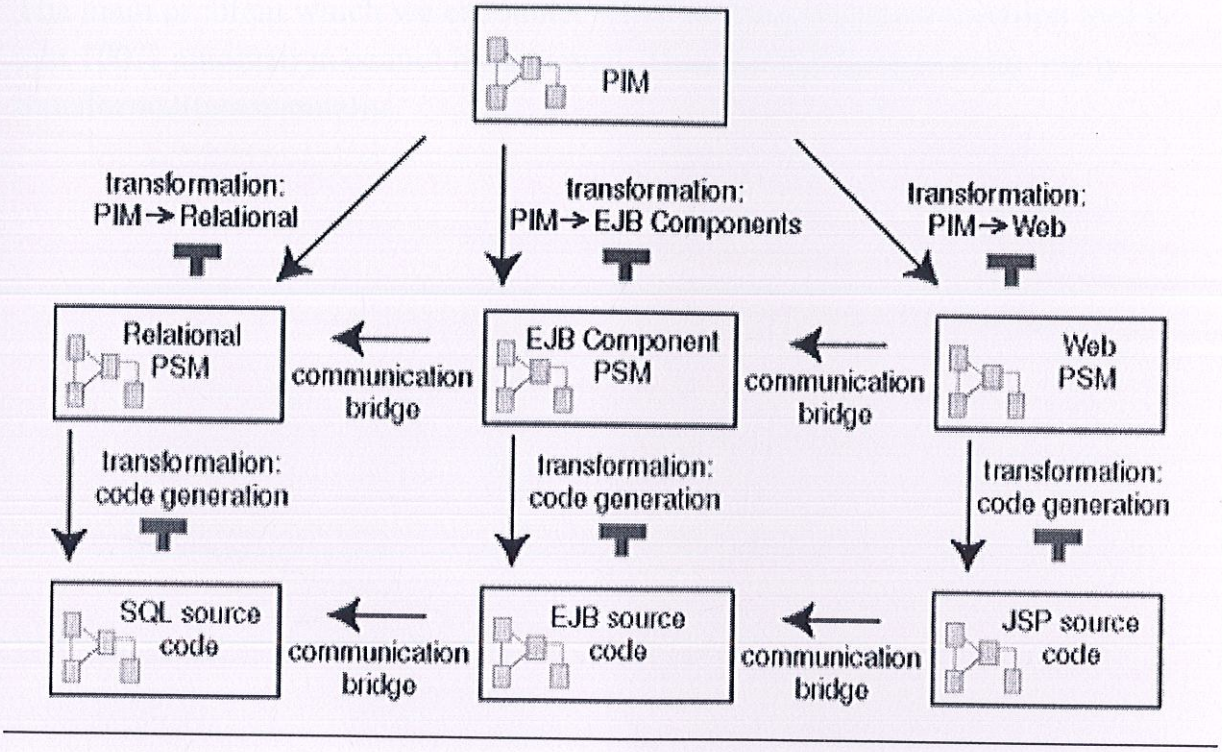
Objective:

To design a tool so that we can convert a Platform Independent Model(PIM) into 3 Platform Dependent Models(PSMs) and to further connect all the 3 PSMs.
This project will help the developer to focus on the details or the requirements of the customer rather than focusing on the details and specifics of the target platforms.
Such an application could be potentially valuable for large firms and small firms where the software required for that firm can be made as per their specifications thus reducing their overall operational cost and time.
The final tool will have a graphical user interface which will help user to give the details of the required class diagram which will be used as an input and through our transformation tool it will be converted into platform specific code.

Scope:

Our project converts the attributes of user desired class diagram into Relational Database and connects each database table with each other on the basis of relation between each table. In future we intend to further transform the user desired class diagram into an EJB component and WEB interface for further user specific requirement for a more user friendly package software.

Diagram:


PIM
transformation:
PIM→Relational
transformation:
PIM→EJB Components
transformation:
PIM→Web
Relational
PSM
communication
bridge
EJB Component
PSM
communication
bridge
Web
PSM
transformation:
code generation
transformation:
code generation
transformation:
code generation
SQL source
code
communication
bridge
EJB source
code
communication
bridge
JSP source
code

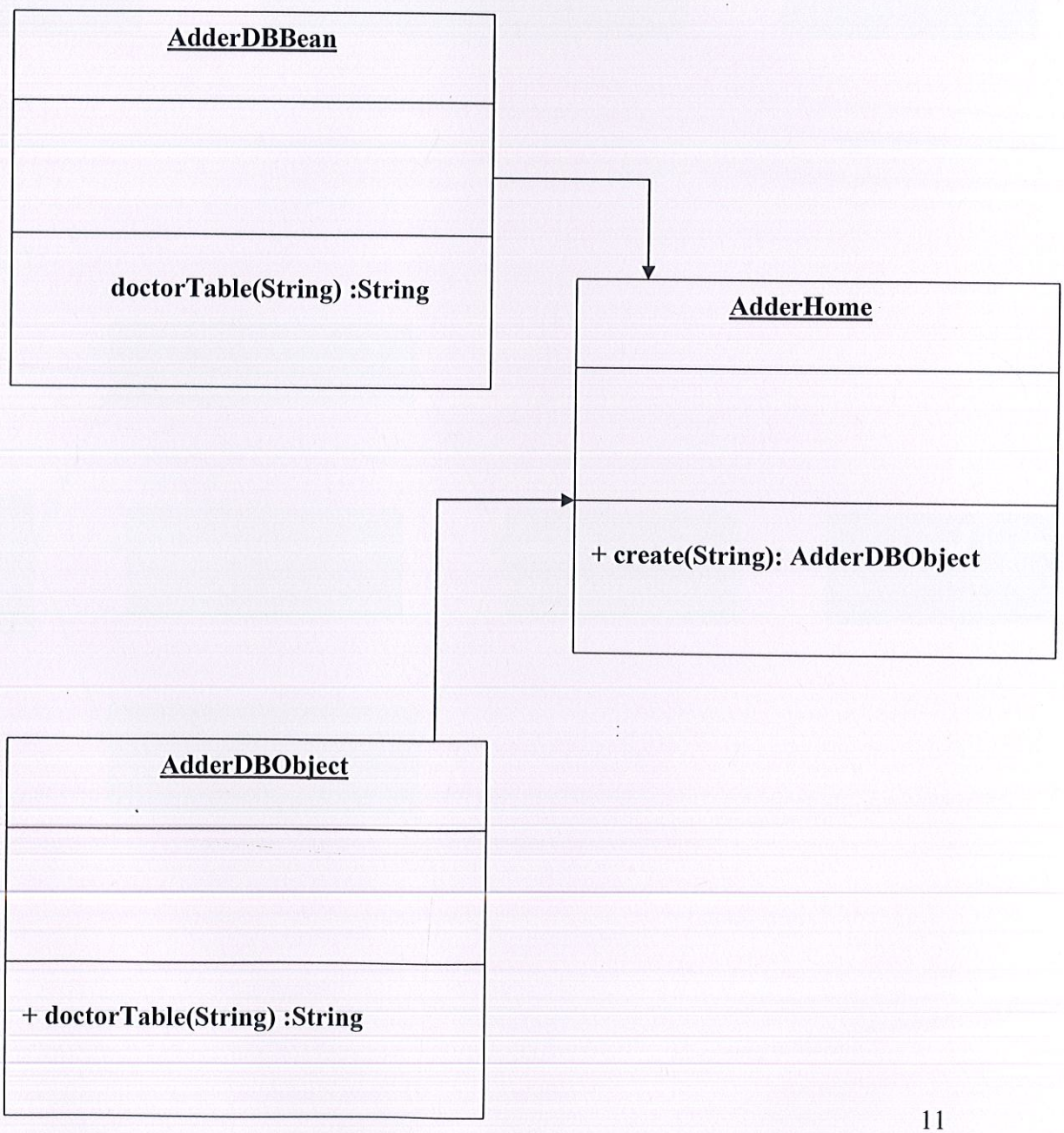
LIMITATIONS:

NOT 100% AUTOMATED:

The main problem which we encounter while making the transformation tool is that 100% automation cannot be achieved. Thus we still have to make many transformations manually.

PROCESS DESCRIPTION

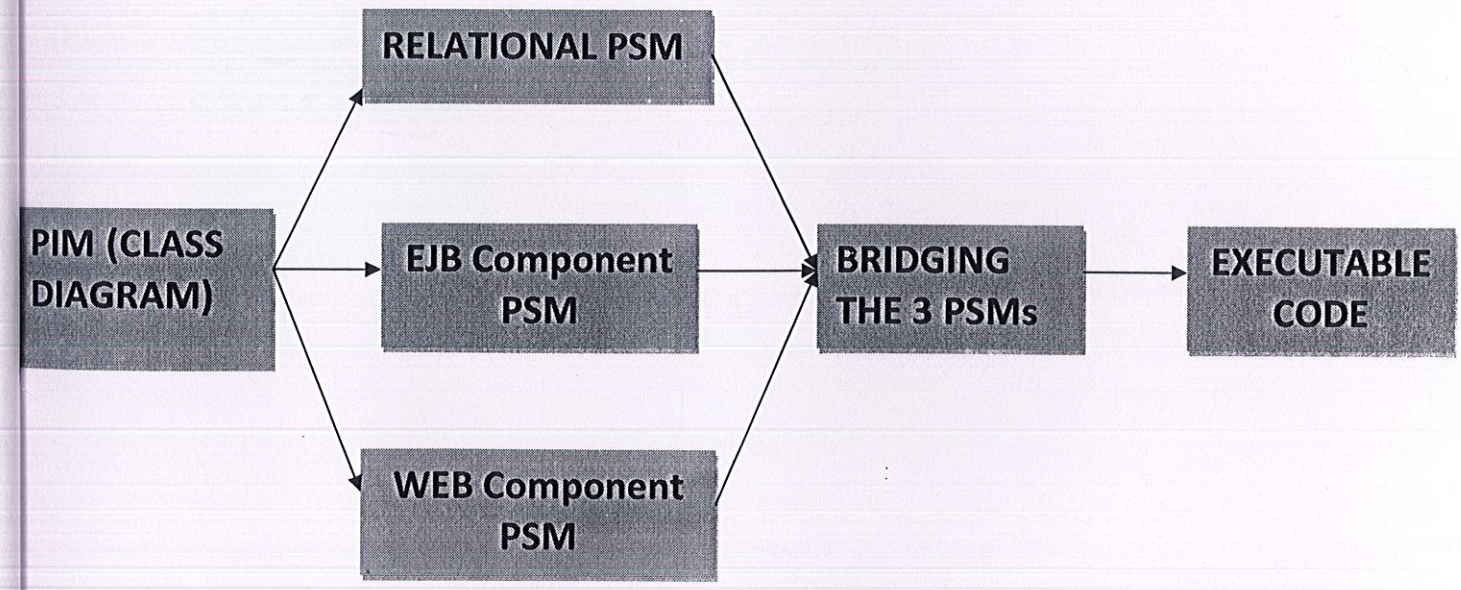
Class Diagram



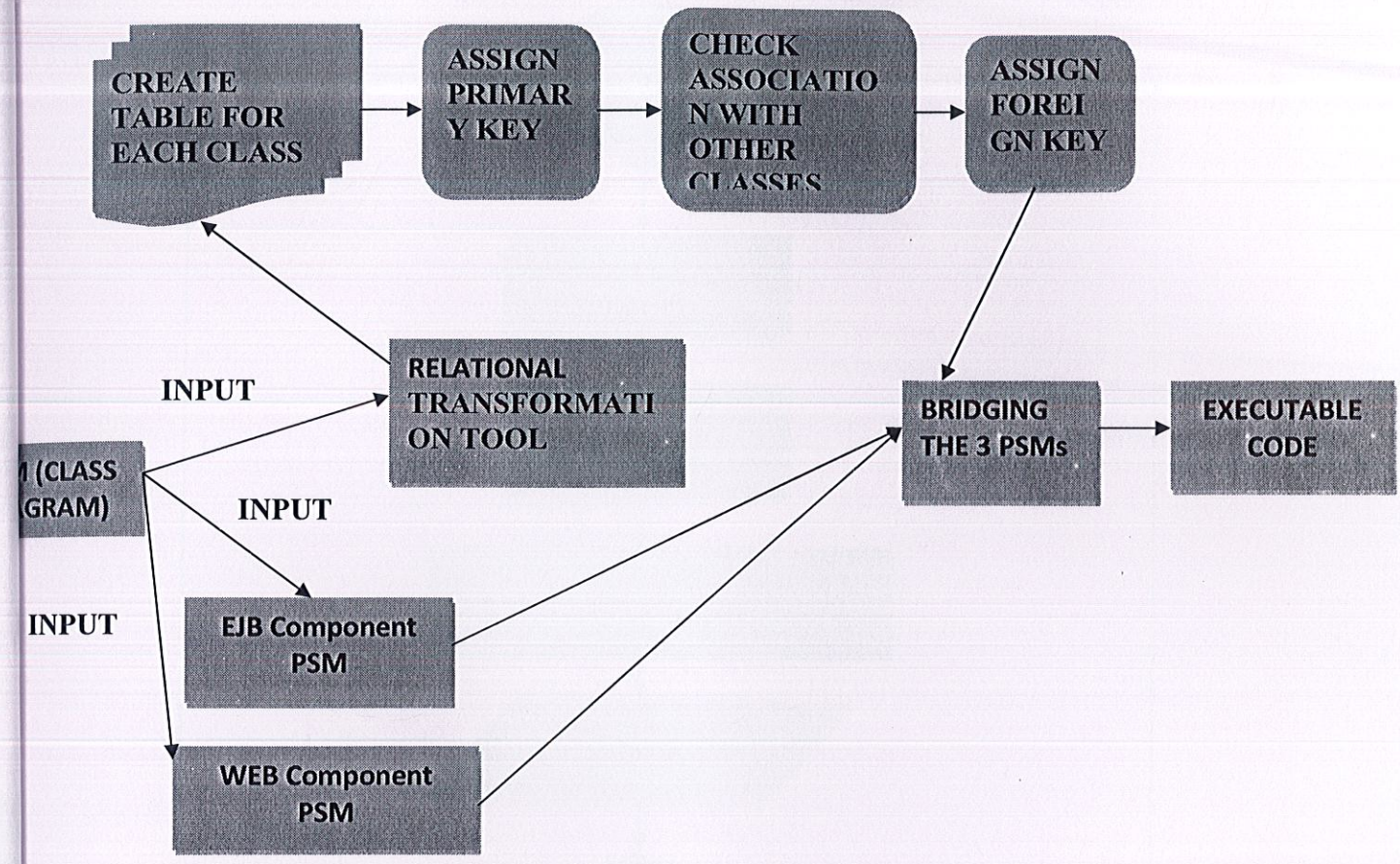
DATA FLOW DIAGRAM:

LEVEL 0:

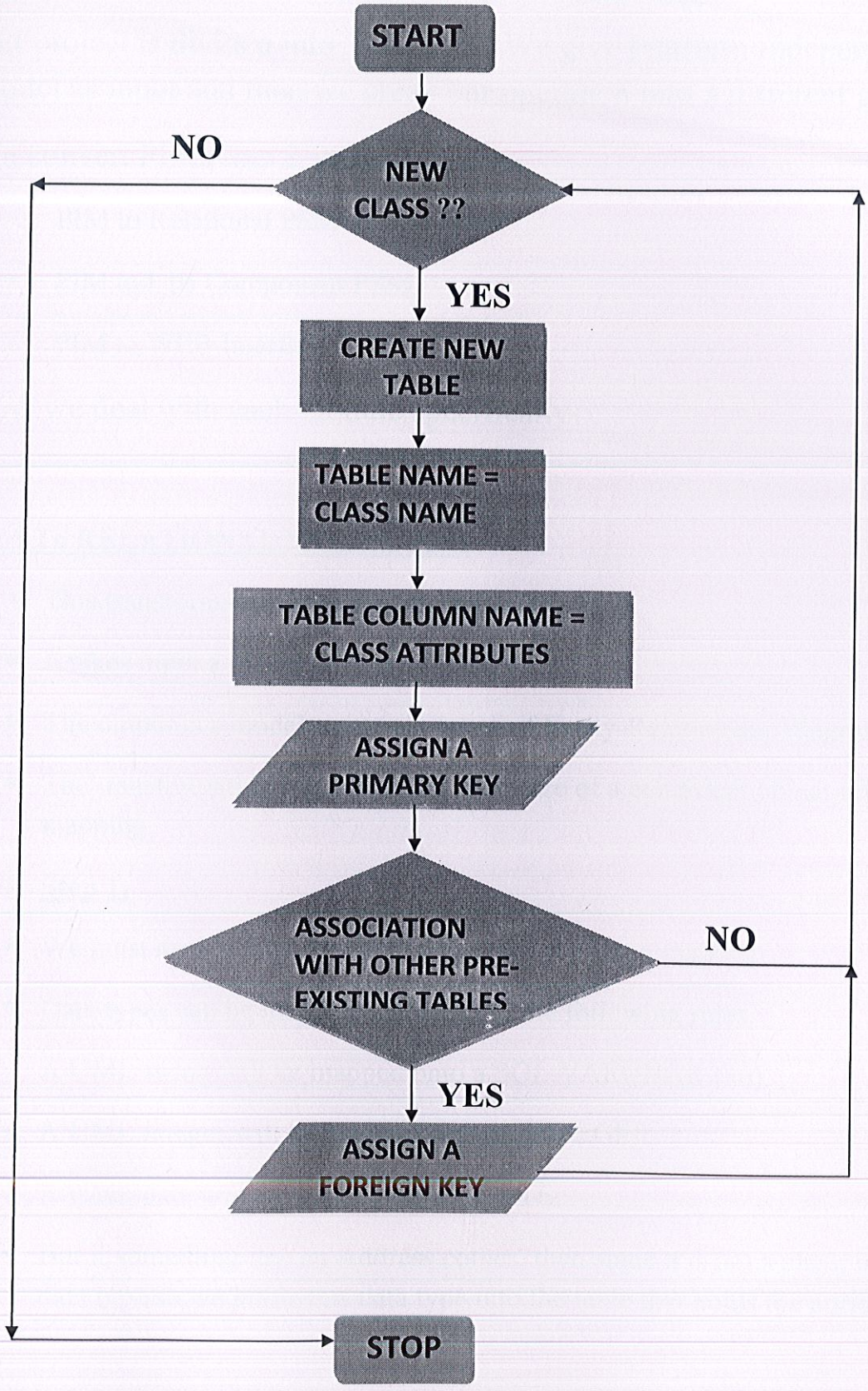
LEVEL 1:



LEVEL 2:


CREATE TABLE FOR EACH CLASS
ASSIGN PRIMAR Y KEY
CHECK ASSOCIATIO N WITH OTHER
ASSIGN FOREI GN KEY
RELATIONAL TRANSFORMATI ON TOOL
INPUT
BRIDGING THE 3 PSMs
EXECUTABLE CODE
(CLASS GRAM)
INPUT
INPUT
EJB Component PSM
WEB Component PSM

FLOWCHART


START
NEW CLASS ??
NO
YES
CREATE NEW TABLE
TABLE NAME = CLASS NAME
TABLE COLUMN NAME = CLASS ATTRIBUTES
ASSIGN A PRIMARY KEY
ASSOCIATION WITH OTHER PRE-EXISTING TABLES
NO
YES
ASSIGN A FOREIGN KEY
STOP

EXPLANATION OF EACH MODULE

Our project is divided into 3 modules. We give Platform Independent Model as input and then we break our operation into 3 different groups.

We convert PIMs into 3 PSMs-

- PIM to Relational PSM
- PIM to EJB Component PSM
- PIM to WEB Interface PSM

Now we deal with each module specifically-

PIM To RELATIONAL MODEL PSM

- This transformation specifies the database.
- It takes input a model written in UML.
- The output is a model written in terms of Entity-Relationship diagrams.
- The transformation rules typically take care of a consistent object-relational mapping.

➢ **Step 1:**

- We must decide how the basic data types are being mapped.
- Data types can be mapped according to the following rules.

✓ A UML string will be mapped onto a SQL VARCHAR (40).

✓ A UML integer will be mapped onto an INTEGER.

✓ A UML date will be mapped onto a DATE.

- But if something like an Address comes, then since it is not a class, but a data type so we inline this data type into the table that holds the attribute.

➢ So, the basic rule is that:

- A UML data type that has no operations will be mapped onto a number of columns, each representing a field in the data type.

Step 2:

- Every class should be transformed into a table.
- All the attributes are fields in the table.
- When the type of the attribute is not a data type but a class, the field in the table should hold a foreign key to the table representing that class.

Step 3:

- Associations in the UML model need to be transformed into a foreign key relation in the database model, by possibly introducing a new table.
- several possibilities for the multiplicities of an association between two classes are:
- The multiplicity at A is zero-or-one.
- The multiplicity at A is one.
- The multiplicity at A is more than one.
- Small changes in the PIM can have a large effect on the relational model.
- For instance, changing the type of an attribute in the PIM from a simple data type to a class means introducing a foreign key in the corresponding table.
- The simple data type can be mapped directly to a column in a table. But if the data type is a class, this class will be mapped to a table itself.
- The column will now have to hold a reference (foreign key) to a key value in that other table.

PIM to EJB Component PSM

- PSM of this transformation is written in a language UML variant.
- This transformation takes as input a model written in UML.
- Outcome is a model written in a UML variant using special EJB stereotypes.

Some of the transformation rules are -:

- For each PIM class, an EJB key class is generated.
- Each PIM class that is not a composite part of another PIM class is transformed into an EJB component and an EJB data schema.
- Each PIM association is transformed into an EJB association within the EJB data schema.
- Each PIM attribute of a class is transformed into an EJB attribute of the mapped EJB data class.
- Each PIM operation is transformed into an EJB operation of the generated EJB Component.

PIM to WEB Interface PSM

- The Web components serve HTML content to the user.
- The PSM for this transformation is also written in a UML variant.
- The transformation takes input a model written in UML.
- The outcome is a model written in a UML variant using special stereotype for the web interface.
- The data types in the Web model define user presentation and interaction details.
- One Web data schema is typically presented to the user in more than one HTML page.

Classes:

AdderDBBean

AdderDBObject

AdderHome

Methods

Method Name : doctorTable()

Class Name : AdderDBBean

Return Type: String

Parameter Type:String

Method Name : doctorTable()

Class Name : AdderDBObject

Return Type:String

Parameter Type:String

Method Name: create()
Class Name : AdderDBObject

Return Type: AdderDBObject

Parameter Type:String

SAMPLE CODE

AdderBean.java

```
package adb;
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import java.util.*;
import java.io.*;
import java.sql.*;
public class AdderDBBean implements SessionBean
{

      public String doctorTable(String query)
      {
            try{
                  Properties db_prop=new Properties();
                  BufferedInputStream db_br=new
BufferedInputStream(new
FileInputStream("C:\\tableprop\\dbprop.properties"));
                  db_prop.load(db_br);

Class.forName(db_prop.getProperty("DB_DRIVER"));
                  Connection
con=DriverManager.getConnection(db_prop.getProperty("DB_UR
L"),db_prop.getProperty("DB_USER"),db_prop.getProperty("DB_
PASS"));
                  PreparedStatement
pstmt=con.prepareStatement(query);
                  int i=pstmt.executeUpdate();
                  return "GOOD TABLE IS CREATED";
            }
```

```
            catch(Exception e)
            {
                System.out.println("AttemptTwoAdderBean: "+e);
                e.printStackTrace();
                return e.toString();
            }
        }
        public void ejbCreate(){}
        public void ejbRemove(){}
        public void ejbActivate(){}
        public void ejbPassivate(){}
        public void setSessionContext(SessionContext sc){}
}
```

AdderHome.java

```
package adb;
import java.rmi.*;
import javax.ejb.*;

public interface AdderHome extends EJBHome
{
      AdderDBObject create() throws
RemoteException,CreateException;
}
```

Created.jsp

```
<%--
   Document   : createdb
```

```
   Created on : May 20, 2010, 1:23:56 PM
   Author     : ssiAdderDBBean
--%>

<%@page
import="javax.naming.*,javax.rmi.PortableRemoteObject,adb.*,ja
va.util.*,java.io.*,java.sql.*" contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <%

      java.util.Enumeration _enum=request.getParameterNames();
     //HERE CREATE QUERY ACCORDING TO SELECTED
FIELDS
      StringBuffer doc_table_temp=new StringBuffer("create table
doctor(");
      NavigableSet ns=new TreeSet();
      while(_enum.hasMoreElements())
      {
         Object obj=_enum.nextElement();
         String
table_col_id=request.getParameter(String.valueOf(obj));
         //out.println("<br>"+obj+" : "+table_col_id);
         //doc_table_temp.append(table_col_id+"");
```

```
            ns.add(table_col_id);
        }
         java.util.Iterator it=ns.iterator();
       //  File file=new File("./doctable.properties");
         //out.println(file.exists());
         Properties prop=new Properties();
                BufferedReader br=new BufferedReader(new
InputStreamReader(new
FileInputStream("C:\\tableprop\\doctable.properties")));
                        prop.load(br);



         while(it.hasNext())
         {
            Object obj=it.next();
           out.print("<br><b>"+obj+"</b>");
            //WITH PROPERTY FILES

doc_table_temp.append(prop.getProperty(String.valueOf(obj)));
            if(ns.higher(obj)==null)
            {
                 doc_table_temp.append(")");
            }
            else
            {
               doc_table_temp.append(",");
            }
         }

                //HERE PRINT THE TABLE QUERY ON WEB PAGE
          out.println(doc_table_temp); //*****************
                //here communication with ejb application

 try {
```

```
        // -------Get an initial context ----------

    Context ctx = new InitialContext();
     Object ref = ctx.lookup("AdderDBBean");

     // Get a reference from this to the Bean's Home interface ----------
-

     AdderHome home =
(AdderHome)PortableRemoteObject.narrow (ref,
AdderHome.class);

     // Create an Adder object from the Home interface ---------------

     AdderDBObject adder = home.create();
     String q=doc_table_temp.toString(); //string query

     String b=adder.doctorTable(q);
     out.println("<center>TABLE CREATED: "+ b+"</center>");
//*********************
}//TRY
catch(Exception e) {
     out.println("*****************************"+e.toString());
   }

      %>
<center>
            <a href="./hospital.jsp">go back</a>
</center>


   </body>
</html>
```

createdPateint.jsp

```
<%--
   Document   : createdb
   Created on : May 20, 2010, 1:23:56 PM
   Author     : ssiAdderDBBean
--%>

<%@page
import="javax.naming.*,javax.rmi.PortableRemoteObject,adb.*,ja
va.util.*,java.io.*,java.sql.*" contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <%

      java.util.Enumeration _enum=request.getParameterNames();
     //HERE CREATE QUERY ACCORDING TO SELECTED
FIELDS
      StringBuffer doc_table_temp=new StringBuffer("create table
doctor(");
      NavigableSet ns=new TreeSet();
      while(_enum.hasMoreElements())
      {
         Object obj=_enum.nextElement();
```

```
            String
table_col_id=request.getParameter(String.valueOf(obj));
            //out.println("<br>"+obj+" : "+table_col_id);
            //doc_table_temp.append(table_col_id+"");

            ns.add(table_col_id);
        }
         java.util.Iterator it=ns.iterator();
       //  File file=new File("./doctable.properties");
         //out.println(file.exists());
         Properties prop=new Properties();
                BufferedReader br=new BufferedReader(new
InputStreamReader(new
FileInputStream("C:\\tableprop\\doctable.properties")));
                        prop.load(br);



         while(it.hasNext())
         {
            Object obj=it.next();
           out.print("<br><b>"+obj+"</b>");
            //WITH PROPERTY FILES

doc_table_temp.append(prop.getProperty(String.valueOf(obj)));
            if(ns.higher(obj)==null)
            {
                doc_table_temp.append(")");
            }
            else
            {
               doc_table_temp.append(",");
            }
        }
```

```
            //HERE PRINT THE TABLE QUERY ON WEB PAGE
        out.println(doc_table_temp); //*****************
            //here communication with ejb application

 try {
      // -------Get an initial context -----------

   Context ctx = new InitialContext();
    Object ref = ctx.lookup("AdderDBBean");

    // Get a reference from this to the Bean's Home interface ----------
-

    AdderHome home =
(AdderHome)PortableRemoteObject.narrow (ref,
AdderHome.class);

    // Create an Adder object from the Home interface ---------------

    AdderDBObject adder = home.create();
    String q=doc_table_temp.toString(); //string query

    String b=adder.doctorTable(q);
    out.println("<center>TABLE CREATED: "+ b+"</center>");
//*********************
}//TRY
catch(Exception e) {
    out.println("*****************************"+e.toString());
  }

     %>
<center>
          <a href="./hospital.jsp">go back</a>
</center>
```

```
</body>
</html>
```

TESTING

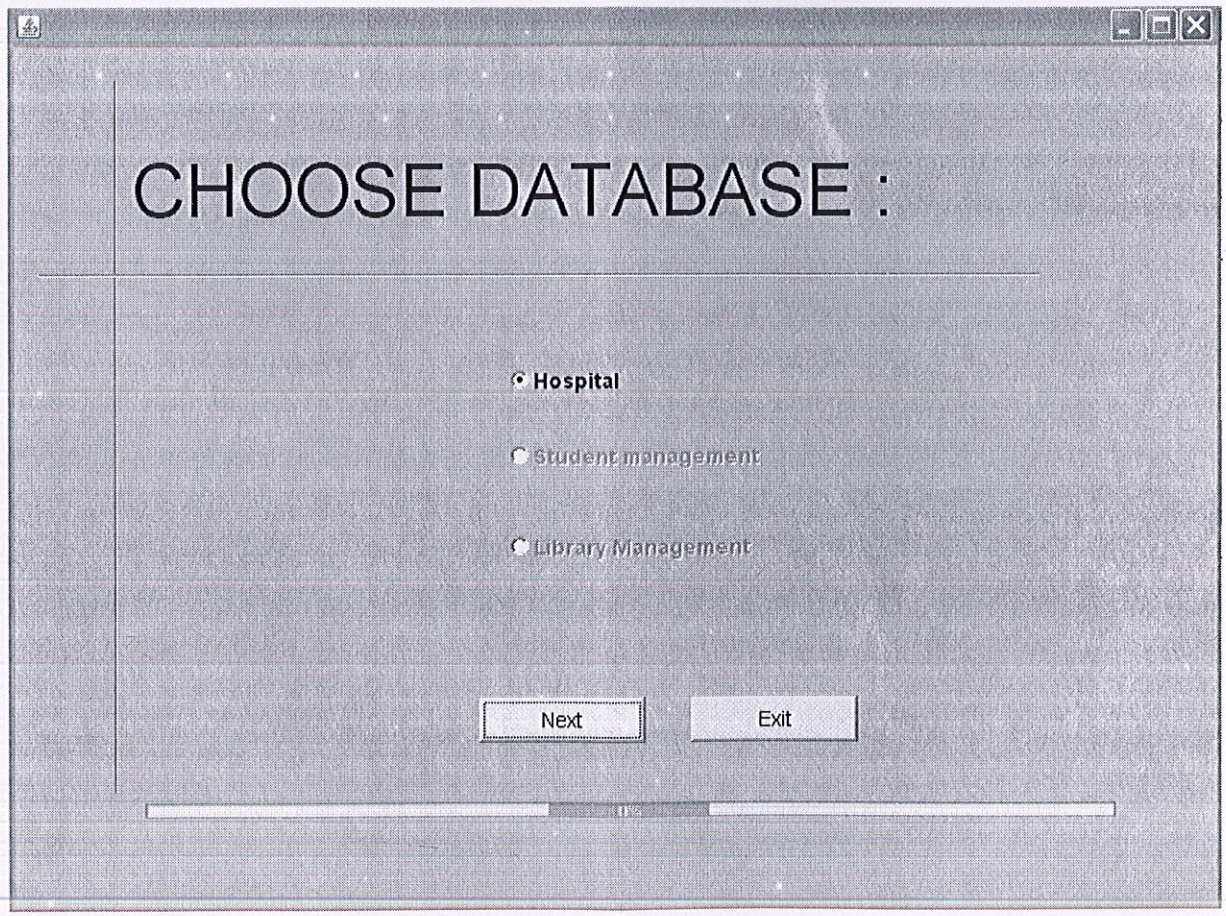
- **BLACK BOX** testing approach was used.
- Some dry data runs were conducted and the resultant output was examined.
- The output was compared with the real desired output thus testing the authentication of our transformation tool.
- We on a random basis changed the attributes to be required in our table, and examined the changes in the database.
- The errors which occurred in formation of the database were further delt with.

RESULT AND CONCLUSION

We were able to successfully implement the modules to .

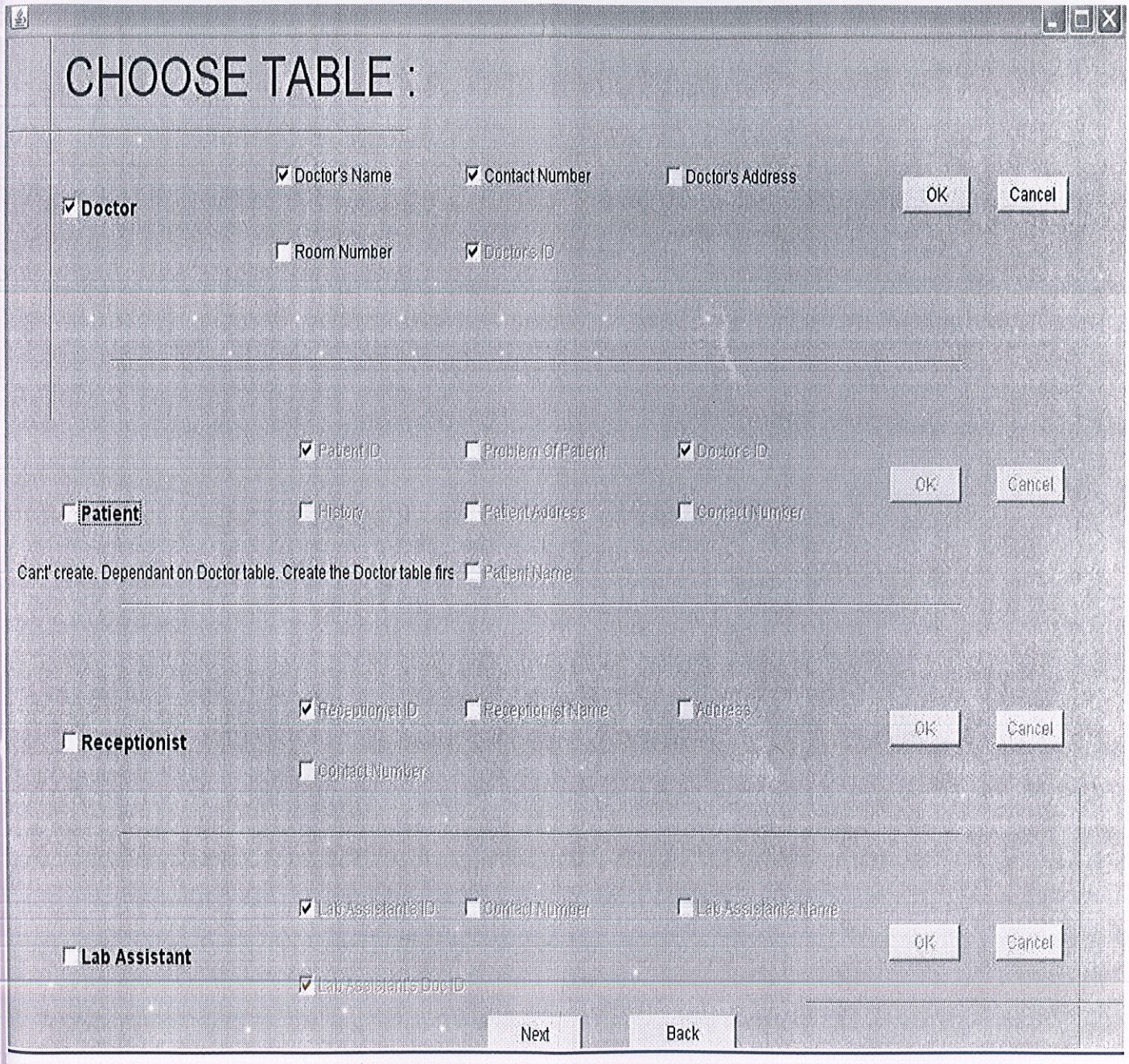
RESULTS:

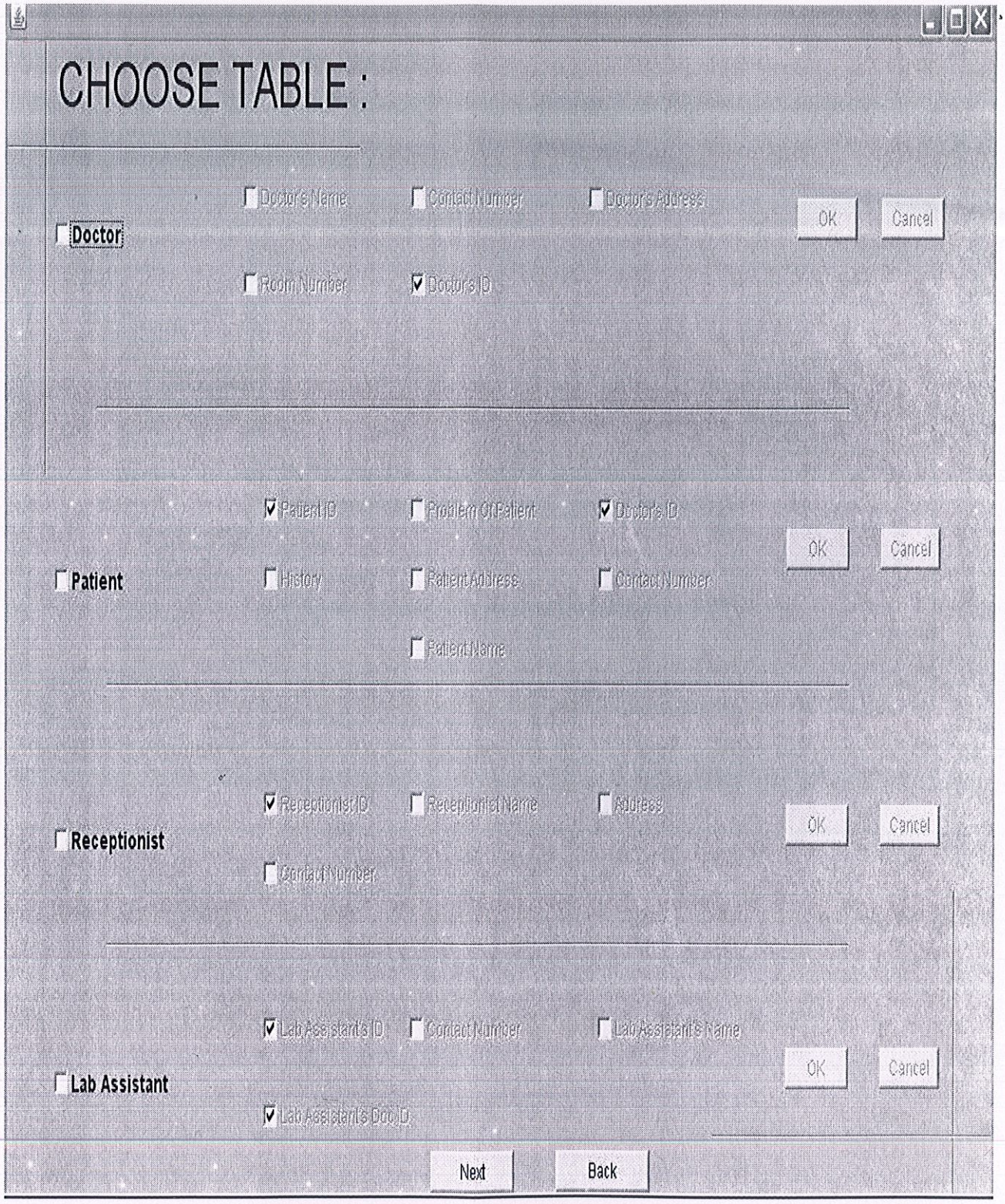
Initial GUI:



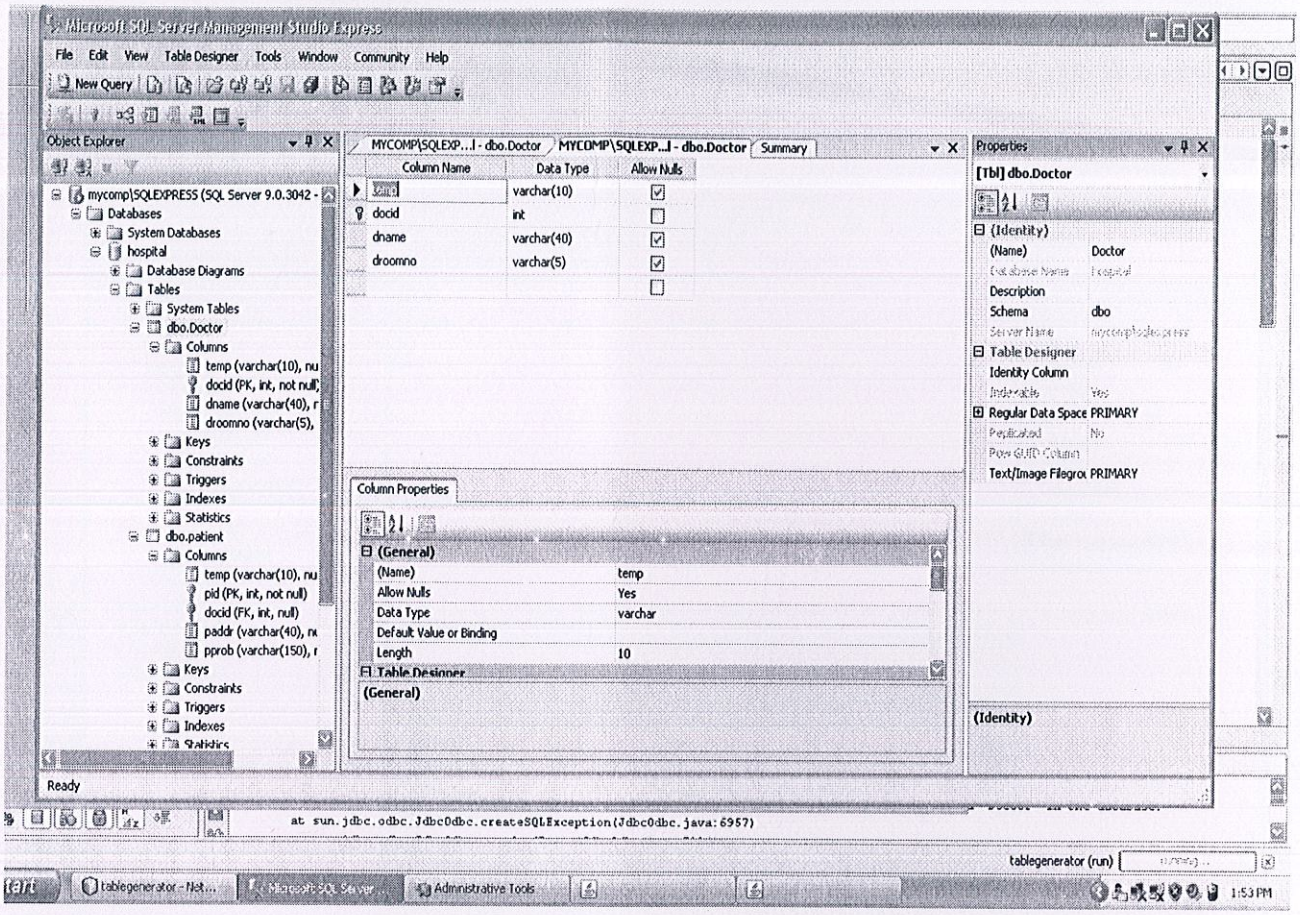
On Pressing The NEXT Button:

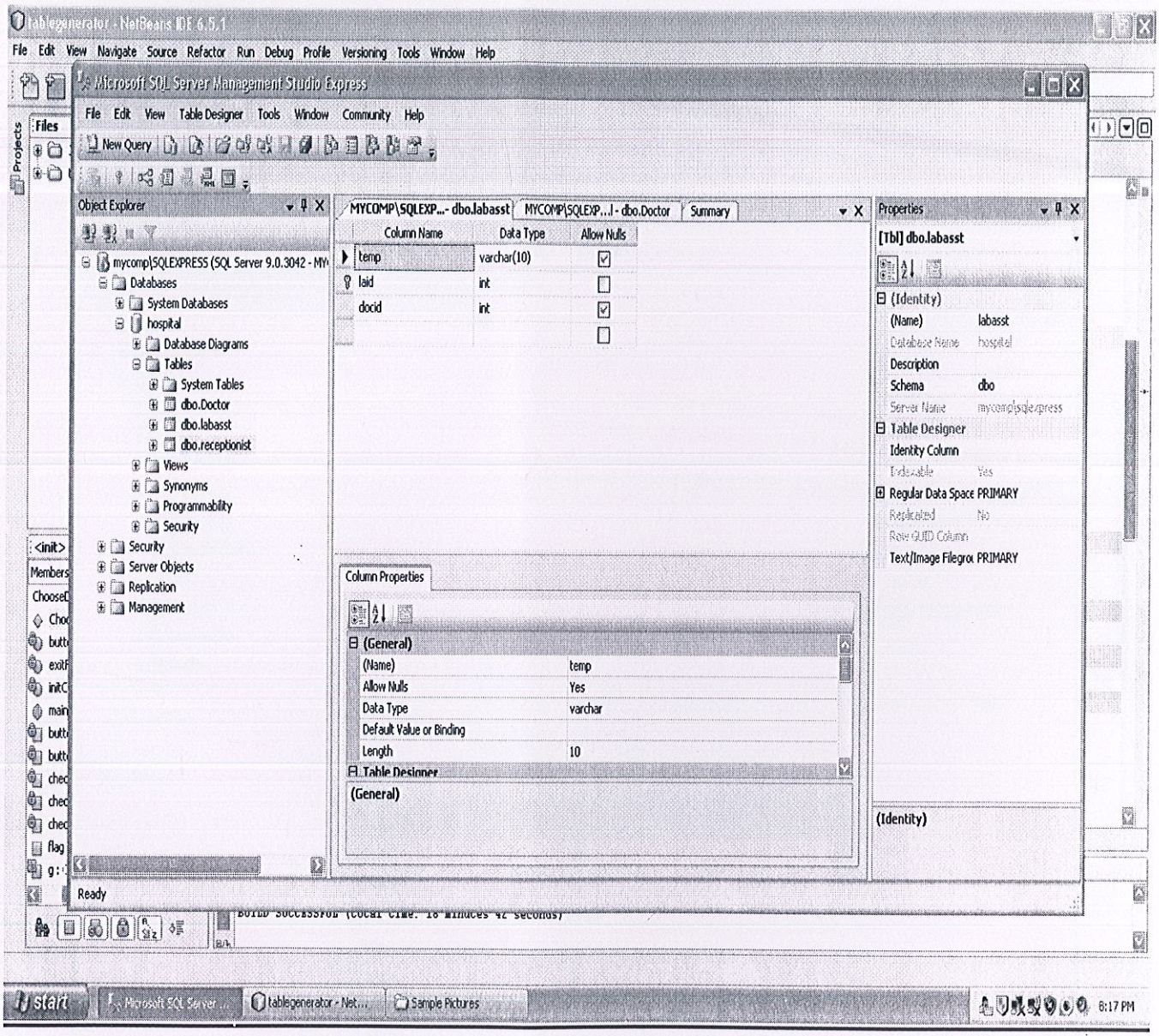
The Details About Each CLASS is asked:

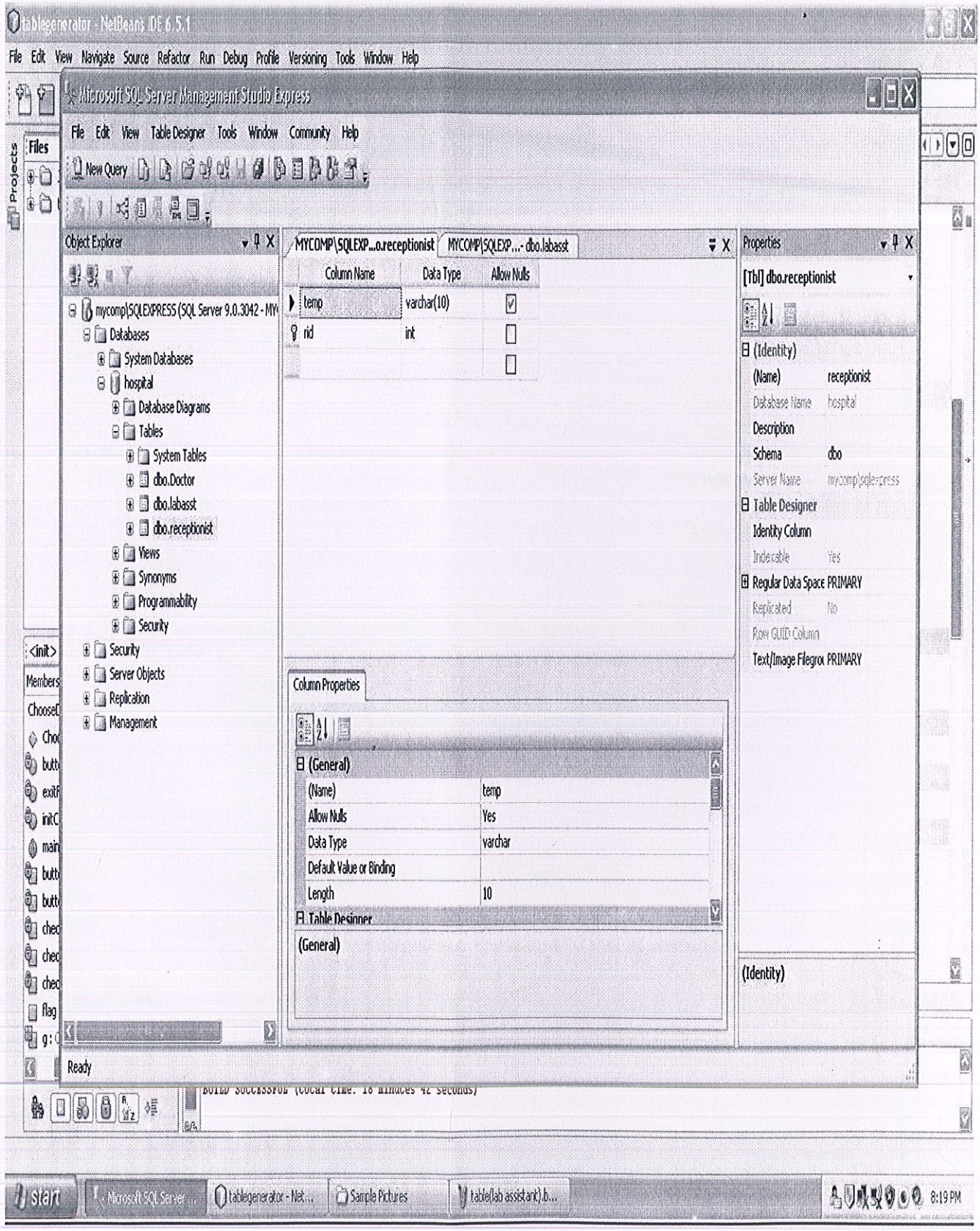


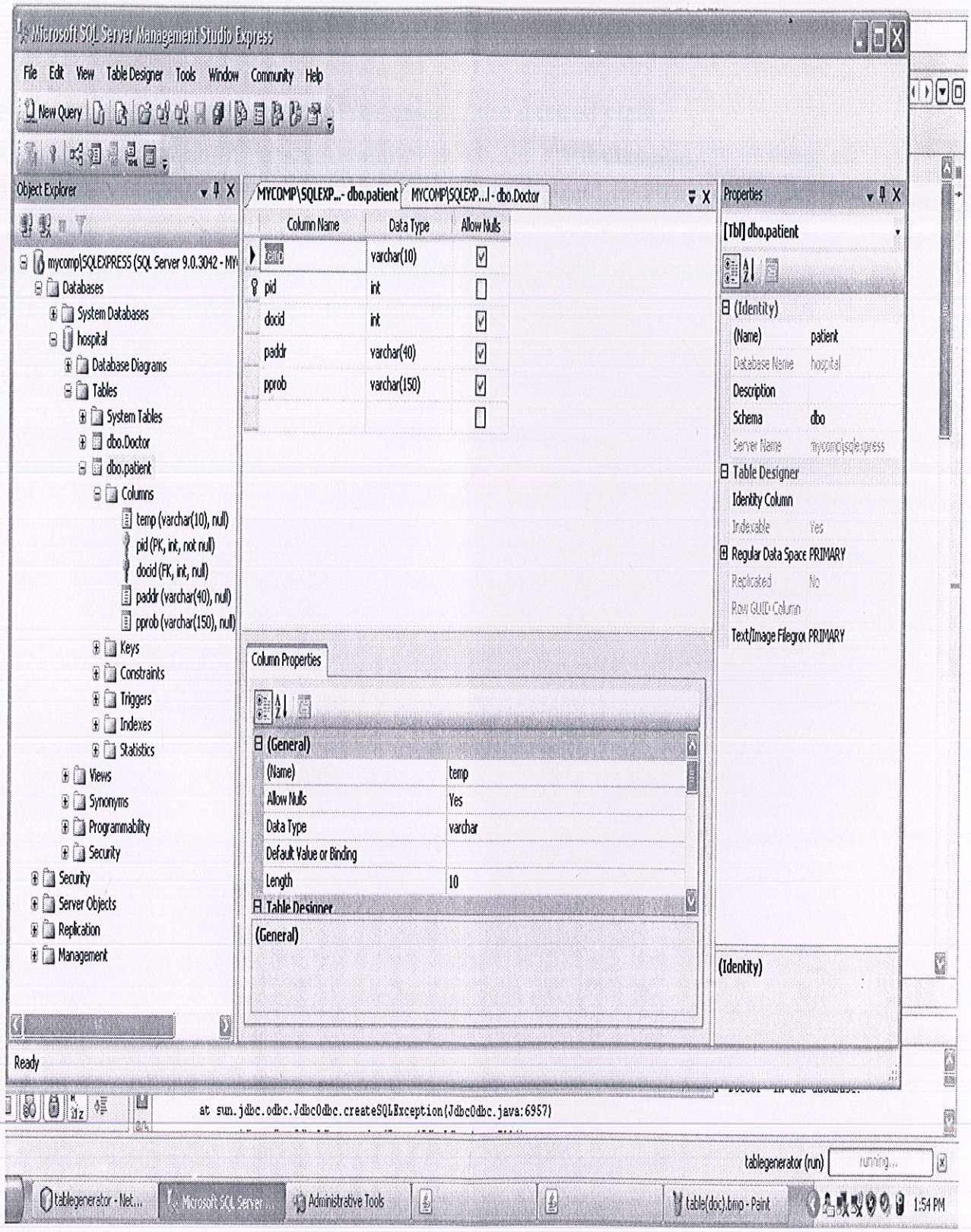

CHOOSE TABLE :
Doctor
Doctor's Name
Contact Number
Doctor's Address
Room Number
Doctor's ID
OK
Cancel
Patient
Patient ID
Problem Of Patient
Doctor's ID
History
Patient Address
Contact Number
Patient Name
OK
Cancel
Receptionist
Receptionist ID
Receptionist Name
Address
Contact Number
OK
Cancel
Lab Assistant
Lab Assistant's ID
Contact Number
Lab Assistant's Name
Lab Assistant's Doc ID
OK
Cancel
Next
Back

DATABASE created at the backend:




tablegenerator - NetBeans IDE 6.5.1
File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help
Microsoft SQL Server Management Studio Express
File Edit View Table Designer Tools Window Community Help
New Query
Object Explorer
mycomp\SQLEXPRESS (SQL Server 9.0.3042 - MY
Databases
System Databases
hospital
Database Diagrams
Tables
System Tables
dbo.Doctor
dbo.labasst
dbo.receptionist
Views
Synonyms
Programmability
Security
Security
Server Objects
Replication
Management
MYCOMP\SQLEXP...- dbo.labasst
MYCOMP\SQLEXP...l - dbo.Doctor
Summary
Column Name
Data Type
Allow Nulls
temp
varchar(10)
laid
int
docid
int
Column Properties
(General)
(Name)
temp
Allow Nulls
Yes
Data Type
varchar
Default Value or Binding
Length
10
Table Designer
(General)
Properties
[Tbl] dbo.labasst
(Identity)
(Name)
labasst
Database Name
hospital
Description
Schema
dbo
Server Name
mycomp\sqlexpress
Table Designer
Identity Column
Indexable
Yes
Regular Data Space PRIMARY
Replicated
No
Row GUID Column
Text/Image Filegro PRIMARY
(Identity)
Ready
start
Microsoft SQL Server ...
tablegenerator - Net...
Sample Pictures
6:17 PM


tablegenerator - NetBeans IDE 6.5.1
File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help
Microsoft SQL Server Management Studio Express
File Edit View Table Designer Tools Window Community Help
New Query
Object Explorer
mycomp\SQLEXPRESS (SQL Server 9.0.3042 - MY
Databases
System Databases
hospital
Database Diagrams
Tables
System Tables
dbo.Doctor
dbo.labasst
dbo.receptionist
Views
Synonyms
Programmability
Security
Security
Server Objects
Replication
Management
MYCOMP\SQLEXP...o.receptionist
MYCOMP\SQLEXP...- dbo.labasst
Column Name
Data Type
Allow Nulls
temp
varchar(10)
rid
int
Column Properties
(General)
(Name)
temp
Allow Nulls
Yes
Data Type
varchar
Default Value or Binding
Length
10
Table Designer
(General)
Properties
[Tbl] dbo.receptionist
(Identity)
(Name)
receptionist
Database Name
hospital
Description
Schema
dbo
Server Name
mycomp\sqlexpress
Table Designer
Identity Column
Indexable
Yes
Regular Data Space
PRIMARY
Replicated
No
Row GUID Column
Text/Image Filegro
PRIMARY
(Identity)
Ready
start
Microsoft SQL Server ...
tablegenerator - Net...
Sample Pictures
table(lab assistant).b...
8:19 PM


Microsoft SQL Server Management Studio Express
File Edit View Table Designer Tools Window Community Help
New Query
Object Explorer
mycomp\SQLEXPRESS (SQL Server 9.0.3042 - MY
Databases
System Databases
hospital
Database Diagrams
Tables
System Tables
dbo.Doctor
dbo.patient
Columns
temp (varchar(10), null)
pid (PK, int, not null)
docid (FK, int, null)
paddr (varchar(40), null)
pprob (varchar(150), null)
Keys
Constraints
Triggers
Indexes
Statistics
Views
Synonyms
Programmability
Security
Security
Server Objects
Replication
Management
MYCOMP\SQLEXP...- dbo.patient
MYCOMP\SQLEXP...I - dbo.Doctor
Column Name
Data Type
Allow Nulls
varchar(10)
pid
int
docid
int
paddr
varchar(40)
pprob
varchar(150)
Column Properties
(General)
(Name)
temp
Allow Nulls
Yes
Data Type
varchar
Default Value or Binding
Length
10
Table Designer
(General)
Properties
[Tbl] dbo.patient
(Identity)
(Name)
patient
Database Name
hospital
Description
Schema
dbo
Server Name
mycomp\sqlexpress
Table Designer
Identity Column
Indexable
Yes
Regular Data Space PRIMARY
Replicated
No
Row GUID Column
Text/Image Filegrox PRIMARY
(Identity)
Ready
at sun.jdbc.odbc.JdbcOdbc.createSQLException(JdbcOdbc.java:6957)
tablegenerator (run)
running...
tablegenerator - Net...
Microsoft SQL Server...
Administrative Tools
table(doc).bmp - Paint
1:54 PM

IMPORTANT FEATURES:

- Database successfully created for each of the desired class.
- Dynamic creation of database achieved, i.e. the attributes and the number of attributes to be needed can be varies successfully and resultant database can be formed.
- Allocation of Primary Key successfully done in each of the table.
- Thus creation of Relational Database has been achieved .